A GPU-Accelerated Network Traffic Monitoring and Analysis System

Wenji Wu, Phil DeMar
Core Computing Division, Fermilab
PO Box 500, MS-368, Batavia, IL 60510, USA
{wenji, demar}@fnal.gov

***Abstract*—Data center networks are evolving toward the use of 40GE between access and aggregation layers, and 100GE at the core layer. With such high data rates, network traffic monitoring and analysis applications, particularly those involved in traffic scrutiny on a per-packet basis, require both enormous raw compute power and high I/O throughput. Many monitoring and analysis tools are facing extreme performance and scalability challenges as 40GE/100GE network environments emerge. Recently, GPU technology has been applied to accelerate general purpose scientific and engineering computing. The GPU architecture fits well with the features of packet-based network monitoring and analysis applications. At Fermilab, we have prototyped a GPU-accelerated architecture for network traffic capturing, monitoring, and analyzing. With a single Nvidia M2070 GPU, our system can handle 11 million+ packets per second without packet drops. In this paper, we will describe our architectural approach in developing a generic GPU-assisted packet capture and analysis capability.**

***Index Terms*—Network monitoring, GPU, multicore, manycore, high-speed networks.**

I. Introduction

Network traffic monitoring & analysis is the process of capturing network traffic and inspecting it closely to determine what is happening in the network. Since the inception of computer networks, traffic monitoring & analysis has been an indispensable tool in network operations & management, capacity planning, and performance troubleshooting. The exponential growth of data intensive applications is driving the need for higher-performance networks. Within the data center, server performance growth, coupled with virtualization capabilities, has fueled 10GbE adoption on the host end. In turn, this has put strain on data center backbone networks, paving the way for deployment of 40GE and 100GE. At such high data rates, network traffic monitoring and analyzing applications, particularly those involved in traffic scrutiny on a per-packet basis, require enormous raw compute power and high I/O throughputs. These applications face extreme performance and scalability challenges.

To date, two major computing platforms have been used for network traffic monitoring & analysis. The first one is dedicated hardware based on NPU (network processor unit) and/or ASIC (application-specific integrated circuits) technologies. The major drawback of this approach is that NPUs and ASICs have poor programmability and poor scalability, and high development costs. The other platform is the general-purpose CPU (central processing unit) that is capable of running many types of applications. There is little doubt that CPUs can be used for network traffic monitoring & analysis. With recent advances in multicore technologies, a single CPU can provide multiple cores to process network traffic in parallel. Researchers have made effective use of multicore systems in network traffic monitoring & analysis. However, we argue that CPU is not the optimum-computing platform for network traffic monitoring & analysis applications at high-performance networks because the CPU architecture does not fit them well.

Recently, GPUs have been widely applied to accelerate general purpose scientific and engineering computing. The massive array of GPU cores offers an order of magnitude higher raw computation power than CPUs. GPU's data-parallel execution model and ample memory bandwidth can effectively hide memory access latency and effectively boost I/O intensive applications with inherent data parallelism. In addition, the CUDA and OpenCL programming frameworks provide GPU with ease of programmability.

At Fermilab, we have prototyped a GPU-accelerated network traffic monitoring & analysis system, which analyzes network traffic on a per-packet basis. In this paper, we will describe our architectural approach in developing a generic GPU-assisted packet capture and analysis capability.

II. A GPU-Accelerated Network Traffic Monitoring and Analysis System

Our GPU-based network monitoring & analysis application runs in user mode, to take advantage of the friendly GPU programming framework (e.g., CUDA or OpenCL). As shown in Figure 1, it consists of four types of logical entities: Traffic Capture, Preprocessing, Monitoring & Analysis, and Output Display.

- Traffic Capture. It captures network traffic and moves them from wire to the CPU domain. Traffic capture aims to capture packets without loss, even at high packet rates.
- Preprocessing. It processes the captured network traffic and copies the packets from the CPU domain to the GPU domain.
- Monitoring & Analysis. It performs network monitoring & analysis with GPUs. We implemented a GPU-accelerated library for network traffic monitoring

and analysis. The library consists of various CUDA kernels, which can be combined in various ways to perform intended monitoring & analysis operations.

- Output Display. Network monitoring & analysis results are displayed or stored.

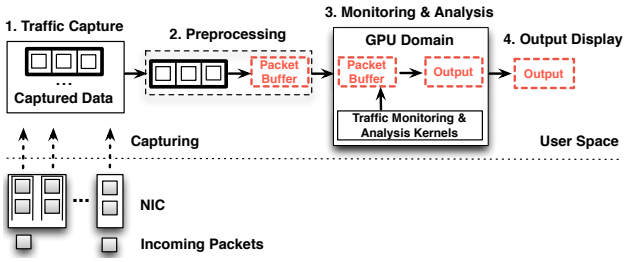

Figure 1 System Architecutre

A logical entity runs on a worker thread. For each type of logical entity, one or multiple worker threads are spawned. On a multicore system, each worker thread is tied to a specific core to maximize overall performance. When the system is in operation, these worker threads run cooperatively to perform an intended task. For the same batch of network traffic, these worker threads run in a pipeline mode with the sequence of "Traffic Capture → Preprocessing → Monitoring & Analysis → Output Display." For different batches of network traffic, these worker threads run in parallel to maximize the overall performance.

Experiments show that our system can achieve extremely high performance. The contribution of our work is threefold:

(1) We demonstrate that GPU can significantly accelerate network traffic monitoring & analysis at high-speed networks. With a single Nvidia M2070 GPU, our system can handle 11 million+ packets per second without packet drops.

(2) We design and implement a generic I/O architecture to move network traffic from the wire to the GPU domain. Our I/O architecture allows network traffic to be captured and copied to the GPU domain without packet loss. Previous studies [1, 2] revealed that the costs of packet capturing derive mainly from three aspects: dynamic per-packet memory allocations and de-allocations of ring buffers, system call overheads, and the cost of moving packets from ring buffers to applications. Our packet I/O engine builds on existing packet-capture techniques, such as pre-allocated large packet buffers, packet-level batch processing, and memory mapping-based zero-copy techniques. Our packet I/O engine works as follows. It divides a NIC receive ring into *descriptor segments*. Each descriptor segment consists of multiple receive packet descriptors (e.g., 1024). The packet I/O engine also pre-allocates a number of empty large *packet buffer chunks*. A packet buffer chunk consists of multiple fixed-size cells, with each cell corresponding to a ring buffer. To capture packets, each descriptor segment in the receive ring will be attached with a pre-allocated packet buffer chunk; each cell in the attached packet buffer chunk is sequentially tied to the corresponding packet descriptor in the descriptor segment. Our packet I/O engine provides operations to allow a user space application to capture packets. The application can access the operations through the *ioctl* interface. Once an attached packet buffer chunk is filled with network packets, it will be "captured" to a user space application. To reduce the capture cost, all packet buffer chunks are mapped into the application's process space. Therefore, an attached packet buffer chunk can be moved to user space via pointer passing; no copying is required. When an attached packet buffer chunk is moved to user space, the corresponding descriptor segment will be attached with a new "free" packet buffer chunk. In the user space, the data in a *captured* packet buffer chunk is finally processed. Subsequently, the chunk will be recycled for future use. Experiments show that our packet I/O engine achieves better performance than Netmap [2] and the I/O engine of PacketShader [1].

(3) We implement a GPU-accelerated library for network traffic capturing, monitoring, and analysis. The library consists of various CUDA kernels, which can be combined in various ways to perform monitoring & analysis tasks. For example, Figure 2 illustrates the mechanism of our packet-filtering kernel. For high-speed network monitoring and analysis, advanced packet-filtering capabilities at wire speed are necessary so that we can monitor and analyze only those packets that are of interest to us. Packet filtering involves establishing a set of packet filter rules. If a packet matches the rules, the packet will be accepted; otherwise, it will be dropped. Because the Berkeley Packet Filter (BPF) [3] is the most widely used packet filter, we use BPF as the packet filter. The richness and expressiveness of the BPF syntax allows us to filter various types of network traffic.

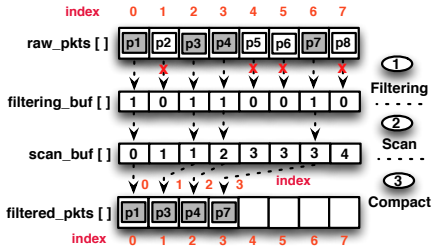

Figure 2 Packet Filtering Kernel

In this poster, we described our architectural approach in developing a generic GPU-assisted network traffic monitoring and analysis capability. Previously, we have been using GPU to analyze network flow records [4]. This project is the next step in our research into network monitoring with GPUs.

Reference:

[1] S. Han et al. PacketShader: a GPU-accelerated software router, In Proc. ACM SIGCOMM'10.

[2] Luigi Rizzo, Netmap: a novel framework for fast packet I/O, In Proc. USENIX ATC'12.

[3] S. McCanne et al. The BSD packet filter: A new architecture for user-level packet capture. In Proc. USENIX Winter 1993.

[4] W Wu et al. G-NetMon: A GPU-accelerated network performance monitoring system for large scale scientific collaborations. In IEEE LCN'11, 2011.